

NISTIR 4685



National PDES Testbed
Report Series

REFERENCE

NIST
PUBLICATIONS



A Conceptual
Architecture
for a
Mechanical Parts
Production System
Based on STEP



QC
100
.U56
#4685
1991

NISTR
OC100
456
4685
1991

National PDES Testbed



**A Conceptual
Architecture
for a
Mechanical Parts
Production System
Based on STEP**

Edward J. Barkmeyer
James E. Fowler
NIST
Spencer P. Magleby
Brigham Young University

U.S. DEPARTMENT OF
COMMERCE

Robert A. Mosbacher,
Secretary of Commerce

National Institute of
Standards and Technology
John W. Lyons, Director

September 27, 1991



Disclaimer

No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied.

This publication was prepared by United States Government employees as part of their official duties and is, therefore, a work of the U.S. Government and not subject to copyright.

Acknowledgement

The National PDES Testbed is funded by the Computer-aided Acquisition and Logistic Support (CALs) Office of the U.S. Department of Defense.

Table of Contents

1 Introduction	1
1.1 Background	1
1.2 STEP Implementation Concepts	1
1.3 Architecture Scope	5
2 System Architecture	7
2.1 Design Engineering Subsystem	8
2.2 Manufacturing Engineering Subsystem	10
2.3 Inspection Engineering Subsystem	13
2.4 Shop Floor Subsystem	16
2.5 Data Repository Subsystem	17
3 STEP Implementation Considerations	22
3.1 Existing Systems	22
3.2 Near-Term Systems	23
3.3 Next Generation Systems	24
4 Summary	27
A Glossary	28
B References	30

List of Figures

1 Data Exchange via Files	3
2 Data Exchange Through Shared Database	4
3 MPPS Subsystems	7
4 Design Engineering Subsystem Components	9
5 Manufacturing Engineering Subsystem Components	12
6 Inspection Engineering Subsystem Components	15
7 Shop Floor Subsystem Components	16
8 Shared Database Components	20
9 Correlation Between AP, AIS, & Software	25



1 Introduction

This document describes an architecture for a manufacturing system using STEP as a primary means for information exchange between software components. The manufacturing system described here is set in the context of mechanical parts production. The architecture presented is intentionally generic. Therefore, the concepts discussed should be largely applicable to other manufacturing domains as well.

1.1 Background

The Standard for the Exchange of Product Model Data (STEP) is an emerging international standard addressing the problems of data exchange and representation for goods produced in a variety of manufacturing enterprises. STEP is a project of the International Organization for Standardization (ISO) Technical Committee on Industrial Automation Systems (TC184) Subcommittee on Industrial Data and Global Manufacturing Languages (SC4). In the United States, the IGES/PDES Organization (IPO) serves to ensure that the requirements of US industry are incorporated into STEP[†]. Participation in both ISO and IPO is voluntary - the National Institute of Standards and Technology (NIST) is active in both organizations. In addition, the PDES, Inc. industry consortium is working with both standards organizations, government agencies such as NIST, and its own member companies to accelerate the development and usage of STEP.

As one aspect of NIST's mission, the agency assists US industry with the assimilation of standards that are used in Computer Integrated Manufacturing. NIST has established the National PDES Testbed specifically to address the development and testing of STEP, and to serve US industry in its use of the standard. Funding for the National PDES Testbed is provided by the Department of Defense's Computer-aided Acquisition and Logistic Support (CALs) Office.

Among the projects planned for the National PDES Testbed is the development of a Product Data Exchange Network [Fre90]. Such a network would link manufacturing sites and research facilities electronically and facilitate STEP validation, implementation, and testing activities. Implementation of various STEP based manufacturing systems, such as that described in [Fow90a], would be a prime candidate for the Network. The architecture presented in this document is equally applicable to an implementation at one physical site and to an implementation distributed over several physical sites.

1.2 STEP Implementation Concepts

Before presenting the details of the architecture, a brief introduction to some of the concepts associated with STEP is required. The reader is encouraged to consult the references for more thorough treatments of these concepts.

[†] PDES is Product Data Exchange using STEP and is used to refer to the US efforts toward the development of STEP.

STEP is a collection of specifications. The individual specifications are known as "Parts"[†] and categorized into resource models, application resource models, application protocols, descriptive methods, implementation methods, and testing methods [ISO1]. The fundamentals of such specifications are briefly examined below with emphasis on implementation concepts.

1.2.1 STEP Resource Models

The STEP specifications which cover the broadest domain of product modeling constructs are grouped in the integrated resource models. The integrated resource models cover such areas as Geometric and Topological Representation [ISO42], Materials [ISO45], Shape Tolerances [ISO47], Form Features [ISO48], etc. All of the integrated resource models are described in the information modeling language EXPRESS [ISO11]. The integrated resource models are meant to be applicable to all product modeling domains. Product modeling constructs from the integrated resource models are the fundamental building blocks used to define Application Protocols.

1.2.2 STEP Application Protocols

Application Protocols (APs) are STEP specifications which define the interpretation of particular elements of STEP within the context of a particular application. An AP defines the specific scope and information needs of the application area, an integrated data model in EXPRESS which meets those needs using elements from the resource models, usage guidelines for the AP, and testing specifications [Pal91]. Examples of APs nearing completion for inclusion in the initial release of STEP include Configuration Controlled Design [ISO203], Mechanical Design using Boundary Representation [ISO204], and Mechanical Design using Surface Representation [ISO205]. There will be many APs defined for STEP. Work is currently underway to ensure that AP development proceeds in a sensible way [Kra91]. APs are the STEP specifications establishing useful and testable product data exchange domains.

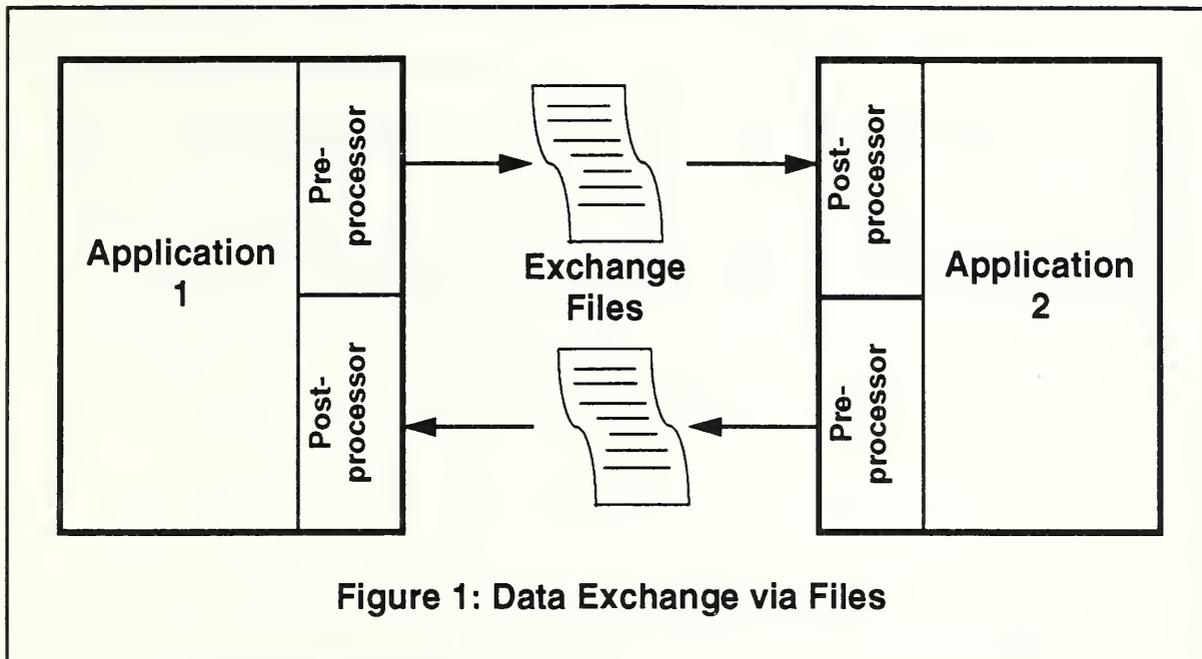
1.2.3 STEP Implementation Methods

There are two implementation mechanisms currently under consideration by the various software developers working to implement STEP. The first is data file exchange. The second is data sharing through a database.

[†] For example the phrase "Part 11 of STEP" means ISO 10303-11 Description Methods: EXPRESS Language Reference Manual [ISO11].

File Exchange

The file exchange method is analogous to that currently employed for IGES [Ree90]. Application software systems exchange information by passing data files from one to the other (see Figure 1). An application software system supports

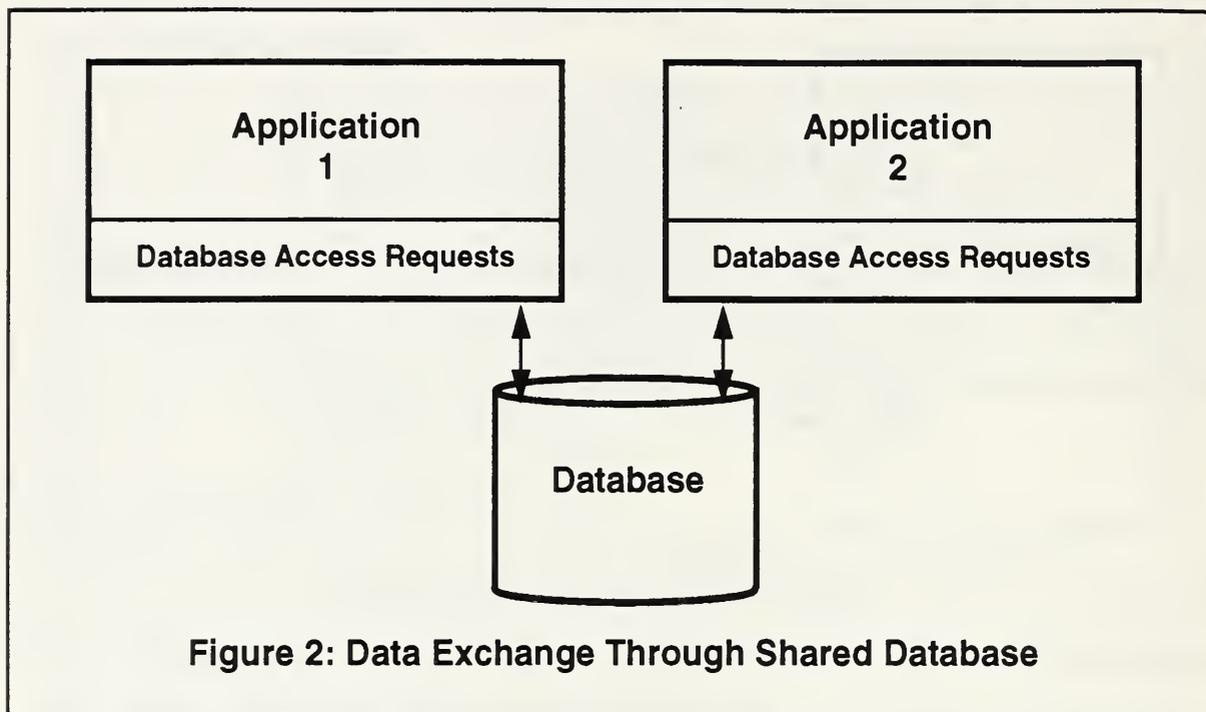


implementation of the standard by having the capability to read data in the standard file format. Reading the file in is known as *post-processing* the data, i.e., the software system is said to have a post-processor which supports the standard. An application software system can also support implementation of the standard by having the capability to output data in the standard file format. Writing the file is known as *pre-processing* the data, i.e., the software system is said to have a pre-processor which supports the standard.

The data file exchange specification for STEP is Part 21 [ISO21]. The exchange specification describes how STEP data is encoded in an exchange file - not how the data is to be interpreted.

Database Sharing

This database sharing method has no analog in the IGES environment. Application software systems exchange information by passing data through a shared database (see Figure 2). The shared database itself contains a



representation of the STEP data to be exchanged. How the STEP data is actually stored in the database is a function of the type of database and could be determined by the user or implementer of the database. There is no standardized scheme for representing STEP data in any type of database. The database management system provides an interface which permits the applications to access and modify the contents of the database. In order to accomplish the information exchange, the developers of the application software systems must agree on (1) a common database system to use, and (2) the STEP storage structure to use. Finally the application developers must know how to use the database interface provided in conjunction with the STEP data storage structure in the database.

The STEP standards organizations are addressing the complexity of database sharing through specification of the STEP Data Access Interface (SDAI) [Fow90b]. This specification will define the functionality for generic interface operations based on the data modeling capabilities of EXPRESS. The specification will be supplemented by a variety of programming language bindings describing how the interface operations are to be used in these languages. The expectation is that application developers, using the interface in conjunction with database software supporting it, would not have to know the underlying details of how STEP data is stored in a database - they would only need to know how the data was described in EXPRESS. There are additional benefits to this approach: applications developers could interface their software with any database supporting the

specification without regard to the type of database. In addition, database developers could optimize the implementation of their systems for STEP in completely proprietary ways.

As with the exchange file specification, the SDAI specification will not describe how STEP data is to be interpreted.

1.2.4 STEP Conformance

Phrases like "supporting the standard" or "supporting the specification" are often used without really defining what is meant. One may also hear phrases such as "STEP-based system" or "STEP application." All of these terms are intended to convey the notion that a particular software implementation is in accordance with some aspect of the STEP standard. This is a purposefully fuzzy definition. The litmus test of whether a software system implements STEP as the standards organizations intend is conformance testing. Only software which has successfully passed a rigorous, independent conformance testing process can be truly recognized as a STEP implementation.

An entire series of STEP specifications focus on the subject of conformance; the General Concepts document [ISO31] and Testing Laboratory/Client Requirements [ISO32] are the first in the series. More specific conformance specifications will follow. In addition, NIST is working towards development of a conformance testing service [Kem91]. The service will establish the procedures by which vendors can obtain certification of their products as conforming STEP implementations.

A primary criterion for conformance will be a product's adherence to one or more STEP application protocols. The intention of the standards organizations is that vendors will supply software products which exchange and interpret product data according to the specifications prescribed in an application protocol. A vendor's product would not be conformant if, for example, it exchanges information according to a conveniently selected collection of geometry and topology from Part 42 [ISO42]. Such an implementation would lack the specific context for interpretation imposed by an application protocol. There would be no criteria against which to measure this product's ability to interpret data it was consuming or for another product's ability to interpret the data the first was supplying. As a result, reliable product data exchange would not be assured.

1.3 Architecture Scope

A manufacturing system architecture describing components and interactions can be quite complex. Describing how STEP would fit in a very general manufacturing architecture might not provide enough concrete information to assist an implementer of a specific system. We expect that initial implementations of STEP will be in focused manufacturing domains paralleling the first set of STEP specifications. The initial STEP specifications largely pertain to mechanical parts production. Initial STEP implementations are likely to be in specific manufacturing functions which could be components of a larger overall

production system. Therefore the context of mechanical parts production is an apt domain for the architecture described in this document as is a focus on those functions where STEP implementation fits in that domain.

The scope of this document then is to identify potential software systems to be used in a mechanical parts production environment that includes STEP APs and examine in a generic sense how appropriate software systems would implement the APs.

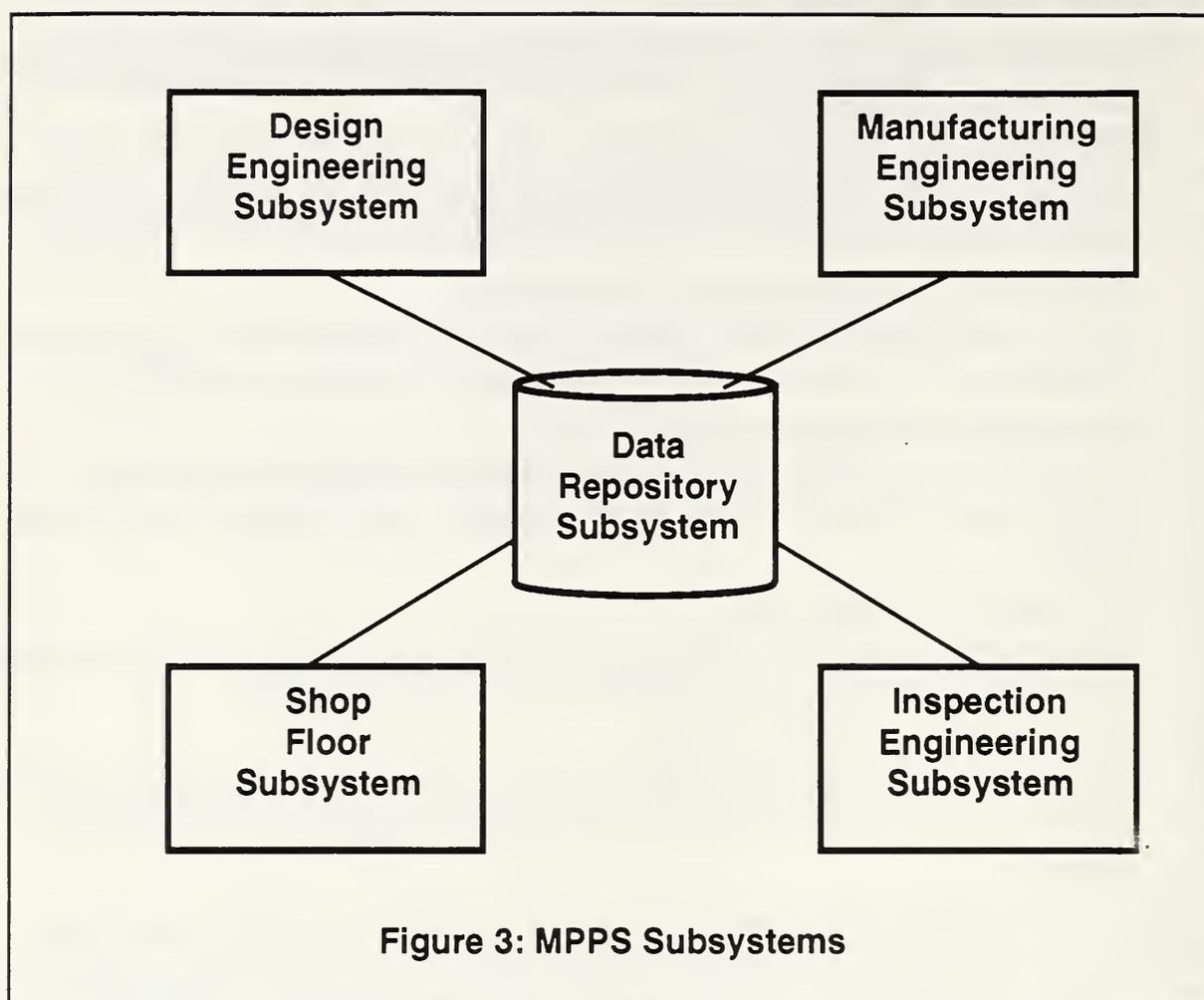
2 System Architecture

We envision a Mechanical Parts Production System (MPPS) comprised of five logical subsystems (Figure 3):

- the Design Engineering Subsystem - incorporates the software needed to generate all of the product description information for the parts to be manufactured by the MPPS;
- the Manufacturing Engineering Subsystem - incorporates the software needed to generate the manufacturing process description information for the parts to be produced by the MPPS;
- the Inspection Engineering Subsystem - incorporates the software needed to generate the inspection process description information for quality assurance of parts as produced by the MPPS;
- the Shop Floor Subsystem - incorporates the software and manufacturing equipment needed to execute the actual production and inspection of parts in the MPPS;
- the Data Repository Subsystem - incorporates those software components providing for the interchange of both STEP and non-STEP information among all of the above "production" systems.

Each of these logical subsystems is actually comprised of several integrated software components. For example, the Manufacturing Engineering and Shop Floor Subsystem include shop-floor planning and control software which provide for the scheduling and execution of the actual manufacturing operations on physical parts. It is entirely likely that each of the logical subsystems will include one or more local databases as well.

In different manifestations of this architecture, the application subsystems (i.e., the Design, Manufacturing, and Inspection Subsystems) will be roughly equivalent within a given manufacturing domain and even across similar domains. Actual MPPSs using this architecture, however, will differ significantly in the nature and choice of hardware and software within each logical subsystem, even within equivalent manufacturing domains. Thus the class of parts which a MPPS can fabricate, and the nature of information exchanged between application systems, will differ according to the APs that the MPPS' implement.



2.1 Design Engineering Subsystem

The Design Engineering Subsystem in this architecture performs all of the functions which generate the information describing the part. In this context the Design Engineering Subsystem provides for the creation, modification, and graphical display of part description information. The part description information must be complete so as to enable the Manufacturing and Inspection Engineering systems to determine which processes to use for manufacturing and inspection. Such part description information could include structure, geometry, features, tolerances, materials, etc.

The Design Engineering Subsystem must be capable of expressing and understanding the product definition in a manner consistent with pertinent STEP APs. The APs that the Design Subsystem will be required to implement are determined by the needs of the other applications in the architecture. Once the pertinent APs are determined, the Design Subsystem must be capable of conforming to all of them.

Design Engineering Software Components

Within the Design Engineering Subsystem there can be numerous tools that deal with various aspects of the design process. These tools may be very complex, providing a vast range of services to a design team. These tools may exchange information with each other during the process of creating a final design.

Ideally, all of the software components in the system would share information using a common STEP AP. Design, however, is a many-faceted problem with multidisciplinary solutions. In practice, different APs will be needed for the integration of different software tools attacking different aspects of the design problem. Every Design Engineering Subsystem implementation will necessarily identify the design tools it comprises, and define the integrating APs accordingly. Since the emphasis of this architecture is not on the design process, but rather on its relationship to the other logical subsystems, the problems of integrating various design tools into a coherent Design Engineering system are largely outside the concern of this architecture.

The Design Engineering Subsystem shown in Figure 4 represents an example

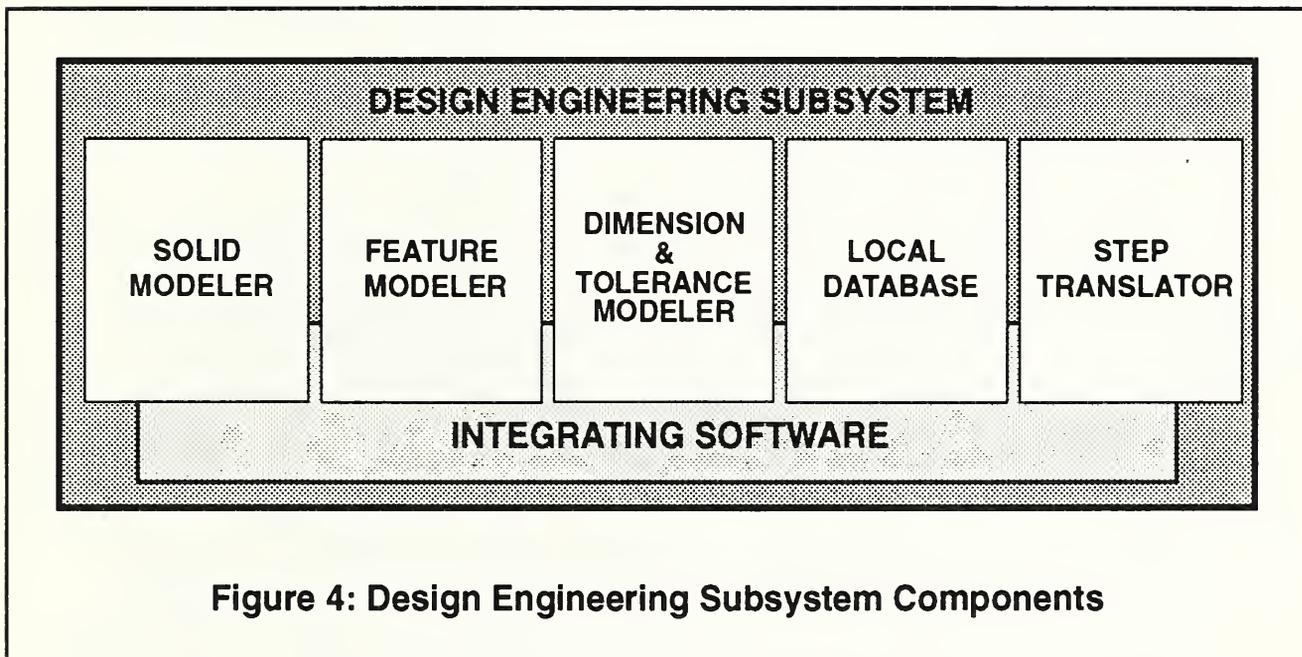


Figure 4: Design Engineering Subsystem Components

decomposition of this system. This decomposition assumes that the product definition is based on a boundary representation geometric definition augmented with features and tolerances. Such representations are found in Part 204, Part 47, and Part 48 [ISO204, ISO47, ISO48]. Other product definition schemes would likely dictate changes in the components but the basic concepts would remain the same.

The Design Subsystem components may be physically combined but can logically be considered as separate entities according to function. With this product definition scheme, the three main design tools of this Design Subsystem are a solid geometric modeler, a feature modeler, and a dimension and tolerance modeler. The solid geometric modeler is used to create the fundamental geometric shape description of a part. The feature modeler allows a designer to

identify features (e.g., bosses, chamfers, etc.) on the solid geometric shape description of a part. The dimension and tolerance modeler allows a designer to establish dimensional tolerances both on the shape and features of a part as well between features themselves. The Design Subsystem could incorporate a local database for intermediate storage of design data. In addition, STEP translation software could be considered as a separate component which performs the functions of exporting design information according to the APs the system supports. Communication among the software components is the responsibility of the integrating software and may or may not use STEP. Thus, the local database subsystem is not required to store design information according to STEP information models.

2.2 Manufacturing Engineering Subsystem

The Manufacturing Engineering Subsystem in this architecture performs all the functions which define the mechanisms and procedures for production of the designed part using a specific collection of shop floor resources. This includes process plans, routing slips, operation sheets, etc. That is, the Manufacturing Engineering Subsystem specifies, at all necessary levels of detail, *how* to make the designed parts.

There are two major classes of information used by the Manufacturing Engineering Subsystem:

- the description of the *products* to be made, and
- the description of the fabrication *resources* available to make those products.

Since this system is to be implemented in a STEP environment, a critical implementation requirement is that there be an AP available defining the information to be shared between the Design and the Manufacturing Engineering Subsystems for the chosen class of parts. This AP represents the "universe of discourse" of the interface between the Design and Manufacturing Engineering Subsystems. In this document we shall refer to this as the Manufacturing Interface AP. All of the information units identified in this AP must be producible by the Design Engineering Subsystem. This is not meant to imply that a single subsystem within the Design Subsystem can support the totality of information required in this AP. Rather, the Manufacturing Interface AP can be seen as the *view* of product information from the Design Subsystem held by the Manufacturing Engineering Subsystem.

In addition to the product description, however, the development of *process* descriptions requires the availability of descriptions of the manufacturing resources available in the shop-floor subsystems, or more specifically, the resource types and their capabilities. There is currently no STEP model for this information base. While work has just started to address this area[†], currently the models used for such information and the structure of the corresponding information repositories must be derived from other sources.

The output of the Manufacturing Engineering Subsystem is several levels of plans for the manufacture of the part, commonly called "process plans." At the top level, the plans are often called routing sheets - they identify the sequence in which the workpiece must be moved to workstations (or workstation types, in the engineering form of the plan), and which "operation sheet" is to be used at each such workstation. The operation sheet is the next level of plan, specifying the handling, fixturing and processing of the workpiece within the workstation, and referencing machine "control programs" for the handling, fixturing and processing operations which are automated. The control program is the lowest level of plan, specifying the detailed machine operation steps required to perform the major operations. Regardless of the degree of automation of the shop-floor subsystems, these data form the conceptual interface between the Manufacturing Engineering Subsystem and the Shop Floor Subsystem. We therefore designate them the Production Interface.

Although there is an existing STEP activity for the development of a process description (or "process plan") resource model, this model is not currently considered to be part of the initial STEP release. Moreover, there are closely related activities and standards, such as APT [ANS87] and BCL [EIA83]. Because the actual Production Interface is intimately connected with the actual fabrication resources available, we consider it to lie outside the immediate scope of this architecture. The use of standards and draft specifications in this area is strongly recommended, but not necessarily a component of a STEP environment[‡]. Interested readers may wish to consult the architecture document produced from NIST's Manufacturing System Integration project [Sen91] for further information regarding the non-STEP aspects of a system architecture^{††}.

Manufacturing Engineering Subsystems

The internal architecture of the Manufacturing Engineering Subsystem can be quite variable and still accomplish the objectives of a MPPS. The decomposition presented in this section should be thought of in a conceptual sense and is not

[†] A new working group (WG8) has been formed in ISO TC184/SC4 to address the definition and use of manufacturing information (e.g., process plans, production resource models) other than product data.

[‡] As the work in WG8 progresses, such specifications may indeed be part of a STEP manufacturing environment.

^{††} The work of ISO TC184/SC5/WG1 is equally applicable to this subject.

meant to constrain the internal arrangement of an actual implementation. This internal architecture decomposition overly simplifies the functions of these subsystems for the purpose of highlighting where STEP fits into the implementation.

Figure 5 diagrams the internal architecture of the Manufacturing Engineering Subsystem. The primary manufacturing software tools are the process planning

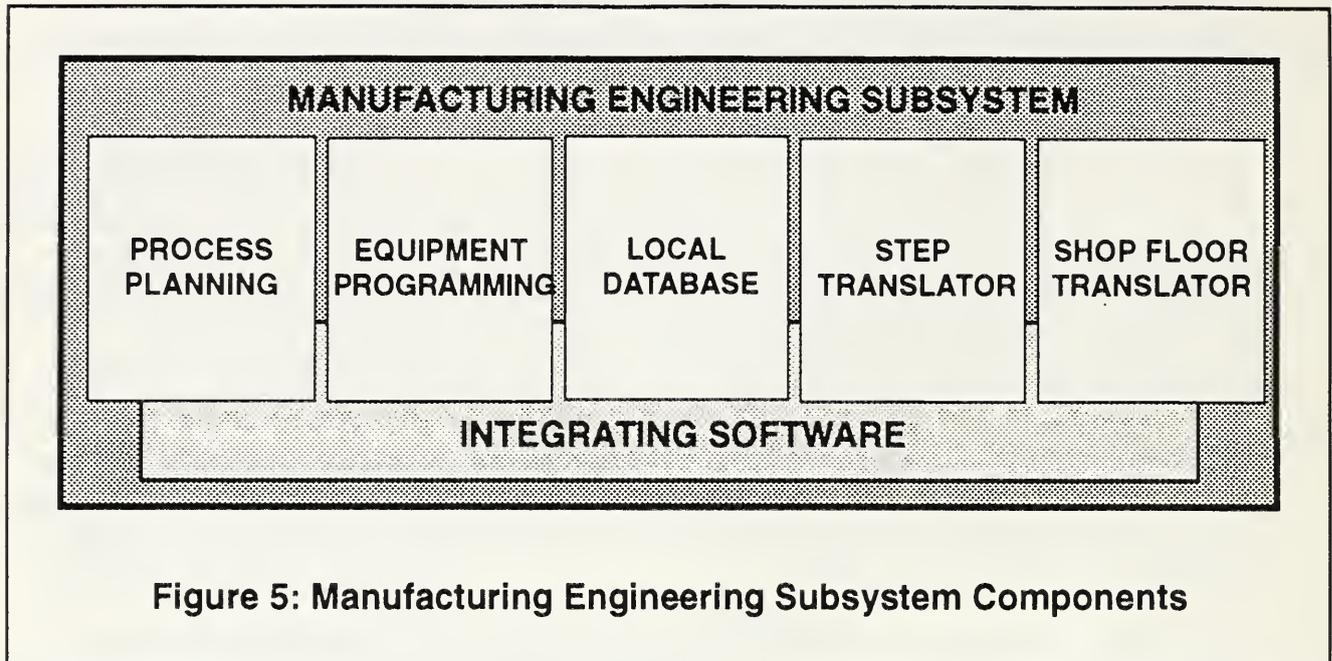


Figure 5: Manufacturing Engineering Subsystem Components

and equipment programming software. The supporting components include a software module which imports data according to the Manufacturing Interface AP and a software module which interfaces with the Shop Floor Subsystem.

In this configuration it would be the responsibility of the STEP translator to obtain STEP data from the Design Engineering Subsystem according to the specifics of the supported Manufacturing Interface AP. This data could then be made available to the rest of the Manufacturing Engineering software components in a format the components could use. A local database could be used for the purpose of storing such data. The shop floor translator is responsible for acquiring descriptions of the shop floor production resources. Such production resource descriptions are needed by the primary Manufacturing Engineering software components. These descriptions too could be stored in a local database. Again, there is no requirement that data is communicated between software components according to STEP information models.

The process planning module requires information describing the production facility resources and the product description. The product description information comes from the Manufacturing Interface AP by way of the STEP translator. Production facility information, i.e., equipment descriptions, tool descriptions and the like, is made available by the shop floor translator. When the process planner has created the manufacturing plan appropriate to the production facility the plan is made available to the equipment programming

module. The equipment programming software would be used to transform the plan instructions into more detailed information, e.g., NC programs, operation sheets, etc., to be used by the shop floor production resources. It is the responsibility of the shop floor translator to make the resulting programs and operation descriptions available to the Shop Floor Subsystem. Thus the output from the shop floor translator constitutes the conceptual Production Interface.

2.3 Inspection Engineering Subsystem

The Inspection Engineering Subsystem in this architecture performs all the functions which define the mechanisms and procedures for inspection of the manufactured product with a specific collection of inspection, measurement, and manipulation resources. That is, the Inspection Engineering Subsystem specifies, at all necessary levels of detail, how to determine whether a given part as-manufactured meets the intended design criteria.

In most ways, the Inspection Engineering Subsystem is completely analogous to the Manufacturing Engineering Subsystem. However, the requirements for inspection processes are different than those for manufacturing process, therefore the two subsystems are considered separate.

Like the Manufacturing Engineering Subsystem, there are two major classes of information used by Inspection Engineering Subsystem:

- the description of the *products* to be made, and
- the description of the inspection *resources* available to evaluate the products during and after manufacture.

In the Manufacturing Engineering Subsystem, we referred to the “Manufacturing Interface AP” which defined the universe of discourse between the Design and Manufacturing Engineering Subsystems. Here we shall refer to an Inspection Interface AP which serves the same function between the Design and Inspection Engineering systems. All of the information units identified in the Inspection Interface AP should be producible by the Design Engineering Subsystem (although, as mentioned earlier, not necessarily by any single subsystem of that system). The Inspection Interface AP is the *view* of the product information from the Design Engineering Subsystem held by the Inspection Engineering Subsystem.

It is possible that there may be considerable overlap between the Inspection Interface AP and the Manufacturing Interface AP. Since the underlying objectives of the two systems, however, are entirely different, one would expect there will be some significant differences in the information required in the two interfaces. Accordingly, the two application protocols should be considered separate.

The output of the Inspection Engineering system is several levels of plans for the inspection of the part. The top level plans call for additions to the routing sheets, identifying the points in production at which the workpiece must be moved to an inspection station and which “inspection plan” is to be used at each such station. The inspection plan is the next level of plan, specifying the handling, measurement and analysis of the workpiece within the workstation, and

referencing machine control programs for the handling and measurement operations which are automated. The control program is the lowest level of plan, specifying the detailed machine operation steps required to perform the major operations. Regardless of the degree of automation of the shop-floor inspection subsystems, these data form the conceptual interface between the Inspection Engineering systems and the Shop Floor Subsystem.

The Production Interface described earlier for the Manufacturing Engineering Subsystem must also include data from the Inspection Engineering Subsystem. This are two reasons for this:

- A given workpiece has a single "routing sheet" which identifies and sequences the routing to both manufacturing and inspection workstations, especially when intermediate inspections are to be performed.
- Some inspection operations may be performed at a manufacturing workstation, during or after manufacturing operations on the workpiece, and therefore become part of the single "operations sheet" at that workstation.

The requirement for Inspection Engineering to transmit measurement evaluation criteria to the Shop Floor Subsystem may necessitate additions to the manufacturing models for process plans and control codes. This is an appropriate modification in order to accomplish the necessary merger of the production and inspection instructions. But, as in the case of Manufacturing Engineering, the definition of a Production Interface itself goes beyond the scope of this architecture.

Inspection Engineering Subsystem Software Components

The Inspection Engineering Subsystem internal architecture is similar to that for the Manufacturing Engineering Subsystem. Both systems import the product definition according to STEP APs, both acquire descriptions of shop floor resources, and both export information to the Shop Floor Subsystem to

accomplish the desired processes. The decomposition illustrated in Figure 6

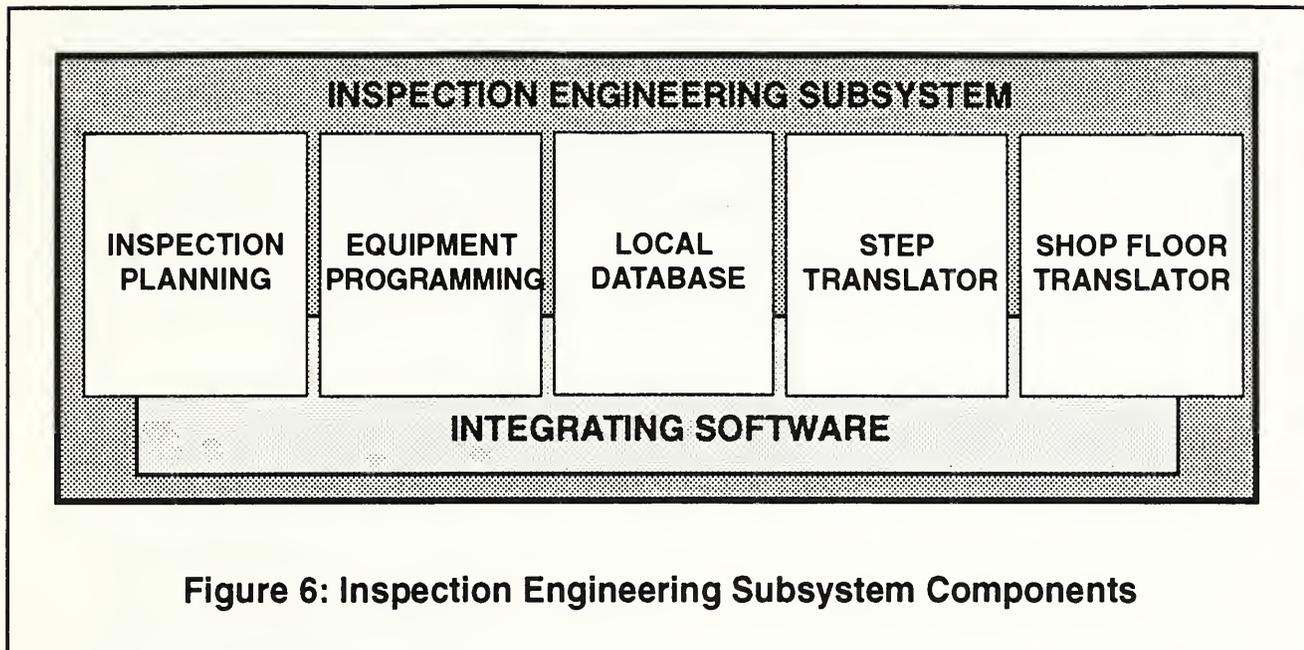


Figure 6: Inspection Engineering Subsystem Components

shows one possible set of subsystems for the Inspection Engineering Subsystem. As with the Design and Manufacturing Engineering decompositions, this internal architecture is to be considered conceptual and not a constraint on an actual implementation.

Here the primary software components are the inspection planning and equipment programming software. The supporting subsystems include a STEP translator which imports STEP data and a shop floor translator. The translator has the responsibility of importing and exporting information from and to the subsystems external to the Inspection Engineering subsystem.

In this configuration it would be the responsibility of the STEP translator to obtain STEP data from the Design Engineering Subsystem according to the specifics of the supported Inspection Interface AP. This data would then have to be made available to the rest of the Inspection Engineering components in a format these components could use. A local database could be used as the intermediate store for such data. As in the Manufacturing Engineering Subsystem, the shop floor translator must acquire the description of inspection resources and make these descriptions available to the primary software components. Additionally, the shop floor translator for this subsystem must have the ability to acquire Production Interface data (e.g., manufacturing equipment routing instructions) so that this information can be updated with inspection instructions. The local database can be used for intermediate storage of these data as well. Again, there is no requirement that data is communicated between software components according to STEP information models.

The inspection planning module requires information describing the production facility resources and the product description. The product description information comes from the Inspection Interface AP by way of the STEP interface.

Production facility information, i.e., inspection equipment descriptions, manufacturing process descriptions and the like, is made available by the Shop Floor translator. The inspection planner creates the inspection plan appropriate for the production processes with the inspection resources available. This plan may necessitate modifications to the manufacturing operation and routing instructions derived earlier in the Manufacturing Engineering Subsystem. The inspection plan is made available to the equipment programming software. As in the Manufacturing Engineering Subsystem, the equipment programming module takes the inspection plan and derives control codes appropriate for automated inspection resources in the Shop Floor Subsystem. The results of the Inspection Engineering activity - routing information, operation instructions, and control codes - form the remainder of the conceptual Production Interface.

2.4 Shop Floor Subsystem

The Shop Floor Subsystem for a MPPS includes the equipment, associated controllers, and software which are used to execute the actual production and inspection of products. Since the MPPS is by definition an environment for mechanical parts, we can describe some of the components typically used in this domain. It is important to note that the actual selection of shop floor resources is intimately related to the specific class of parts to be produced, the APs supporting information interchange between application subsystems, and the capabilities of the application subsystems themselves.

Shop Floor Subsystem Components

Figure 7 illustrates typical components envisioned for the production capabilities

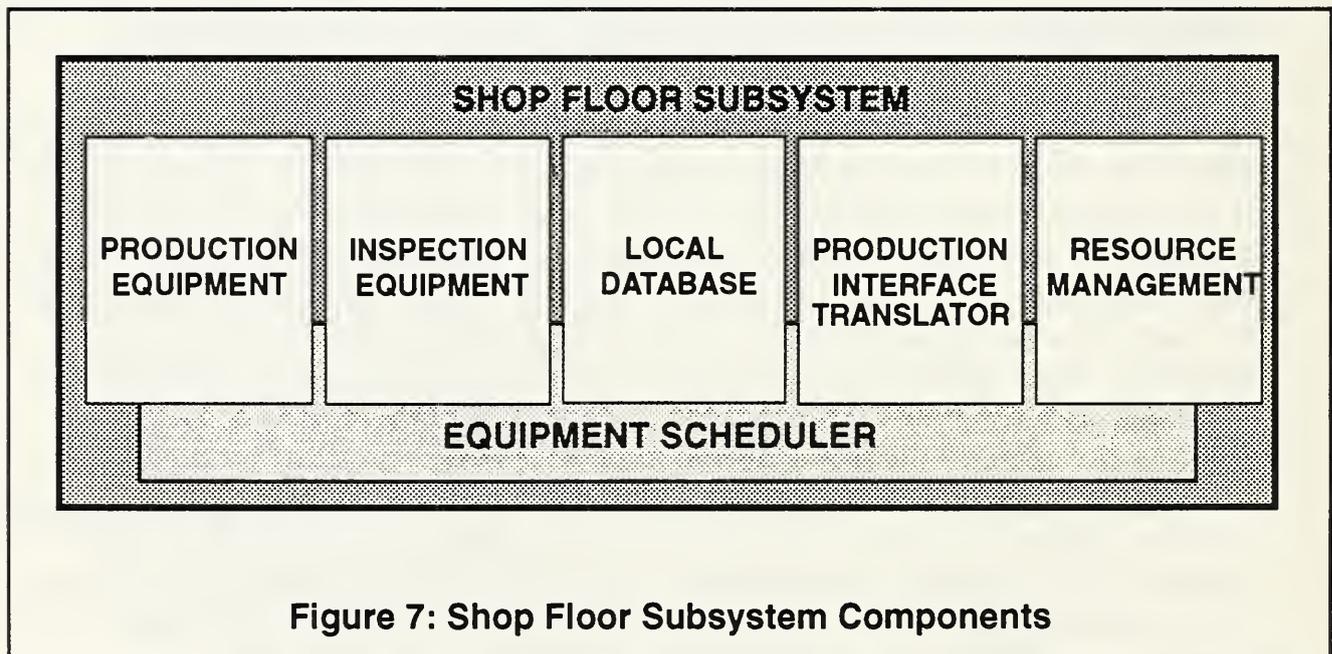


Figure 7: Shop Floor Subsystem Components

of an MPPS. This decomposition of the Shop Floor Subsystem is given at a high level of abstraction and is only meant to give the reader a sense of what

constitutes a mechanical parts production environment. An implementation of an actual MPPS might include only a few of these components or a much more complex arrangement.

The primary components of the Shop Floor Subsystem are the production and inspection equipment. Production equipment resources could include machining centers. Machining centers come in different varieties (e.g., milling, turning, grinding, etc.) but their common purpose is material removal - i.e., processing raw stock into the desired form. Modern machining centers are computer controlled. Inspection equipment resources could include coordinate measuring machines (CMMs). CMMs operate like machining centers but instead of using a tool to remove material from a workpiece they use a sensitive probe to determine the dimensions of a part under inspection.

Machining centers and CMMs receive the computer codes which drive them from the equipment programming modules of the Manufacturing and Inspection Engineering Subsystems. This data, which we described as one aspect of the Production Interface, is acquired for the Shop Floor Subsystem by the production interface translator. The raw control codes generated by the equipment programming modules will usually need to be post-processed and possibly reformatted before the data is ready for consumption by the machine controllers. Here, such processing is handled by the production interface translator. In the future shop floor controllers may be able to handle this data without post-processing.

Aside from the production interface translator, other supporting software components could include equipment scheduling software, resource management, and a local database. The scheduling software optimizes the use of available machine resources given the production requirements generated by the engineering application subsystems. Resource management for raw stock, tools, and the like must also be coordinated with scheduling software to achieve effective production control. Again, as with the other subsystems, a local database is available for intermediate storage of whatever data is necessary for the software components internal to the Shop Floor Subsystem. Certainly for complex fabrication facilities, more complex supporting software components will be required.

2.5 Data Repository Subsystem

The Data Repository Subsystem is responsible for providing storage and access to the information exchanged between subsystems. The information stored by the Data Repository Subsystem will include both STEP and non-STEP data. As described in this architecture, STEP data will be shared between the application subsystems. Non-STEP data will be exchanged between the Manufacturing Engineering and Shop Floor Subsystems and between the Inspection Engineering and Shop Floor Subsystems. Two mechanisms are considered for implementation of a Data Repository Subsystem: file exchange and shared database. Selection of one or the other of the mechanisms has certain ramifications in the implementation of the translator software components described for each of the

subsystems. Additionally, the physical communication network available for electronically linking the subsystems influences whether or not the subsystems can be geographically distributed.

File Exchange

A file exchange mechanism for communicating STEP data between systems is most straightforward. In this case, exchange files containing STEP data according to the specifics of the Manufacturing and Inspection Interface APs would be pre-processed by the Design Engineering system and post-processed by the Manufacturing Inspection Engineering systems respectively. Here the STEP translation software within the Design Engineering Subsystem is responsible for transforming data internal to Design Subsystem components into the STEP data prescribed by the APs and exporting that data in the STEP file format [ISO21]. Conversely, the STEP translation software within the Manufacturing and Inspection Engineering Subsystems is responsible for importing the files and then transforming the STEP data contained within into data internally useful to the systems. One could consider the collection of files under the management of the host computer file system to be the Data Repository in this case[†].

The only difference in using file exchange for non-STEP data is that the format and contents of the files would not be according to STEP. For example, in sending NC program information from the Manufacturing Engineering Subsystem to the Shop Floor Subsystem, the BCL format could be used. For some information exchanges, there may be no accepted specification for the format and content of data - these would then be determined by the requirements of the software components involved.

A shared database mechanism is more complex than file exchange. The next section explores such an implementation. However the reader is urged to consult the PDES, Inc. Data Sharing Architecture document [PDE91] for a more detailed treatment.

Shared Database Components

Figure 8 illustrates components of a Data Repository Subsystem for a shared database implementation. The figure shows both the data repository system and applications linking with the repository - the repository components are differentiated by the cross-hatching in the illustration. In the context of this document the applications of interest are the Design, Manufacturing, and Inspection Engineering systems. These applications communicate STEP data with each other by storing STEP data in the repository and retrieving this data from the repository. The specific data to be stored and retrieved is defined by the APs - in this case, the Manufacturing and Inspection Interface APs. The applications

[†] While a host computer's file system could serve to manage a collection of data files, a more sophisticated approach would impose a configuration control system on the files. At the other end of the spectrum, one could consider a rack of magnetic tapes containing data files as a data repository as well.

communicate with the STEP Data Repository via the STEP Data Access Interface (SDAI). This component essentially hides the details of the STEP data storage from the applications.

The SDAI operates in conjunction with the underlying databases and Information Resource Dictionary System (IRDS) [ANS90] components. Three database components are shown in Figure 8; this is intended to convey the notion that the repository may be comprised of multiple databases - not that there is a specific requirement about the number of databases. The databases provide the physical storage of STEP data in whatever format is appropriate. The database storage format is entirely implementation dependent, thus it is outside the scope of this document and indeed outside the scope of STEP. The SDAI software communicates with the databases through interfaces which are specific to each database implementation.

The STEP data prescribed by the Manufacturing and Inspection Interface APs can be distributed across the databases in the repository. The IRDS serves to keep track of where data is stored. The SDAI software communicates with the IRDS in order to correctly respond to applications' requests for data. Additionally, the IRDS can manage the computer representations of the STEP APs as well (not the data associated with the APs but the AP information models themselves). This *meta-data* is made available to the IRDS through the EXPRESS compiler[†]. The EXPRESS compiler processes the AP information models (often referred to as AP schemas) into computer data structures which can then be used by the IRDS.

As with the description of the internal architectures of the other logical systems, it is important to remember that what has been presented here is only one possible decomposition. More complex implementations could involve remote access to databases, through the SDAI, which are geographically distributed. A simpler realization could involve only a single database, obviating the services of the IRDS, but still supporting the SDAI. The key architectural requirement is that both the underlying repository in a shared database implementation and the applications communicate through the SDAI.

Using a shared database implementation for the Data Repository Subsystem thus implies that the STEP translation software described earlier for the Design, Manufacturing, and Inspection applications is quite different than that described for a file exchange implementation. Whereas the STEP interface software for file exchange plays the role of pre- or post-processor, here it uses the SDAI provided by the Data Repository to make internal data available to the other applications. The STEP translation software operation must therefore conform to both the specifics of the APs and the specifics of the SDAI[‡].

[†] Recall that STEP information models are described using the EXPRESS language [ISO11].

[‡] The SDAI specification itself is still evolving in the STEP community. It has tentatively been identified as Part 22 of STEP.

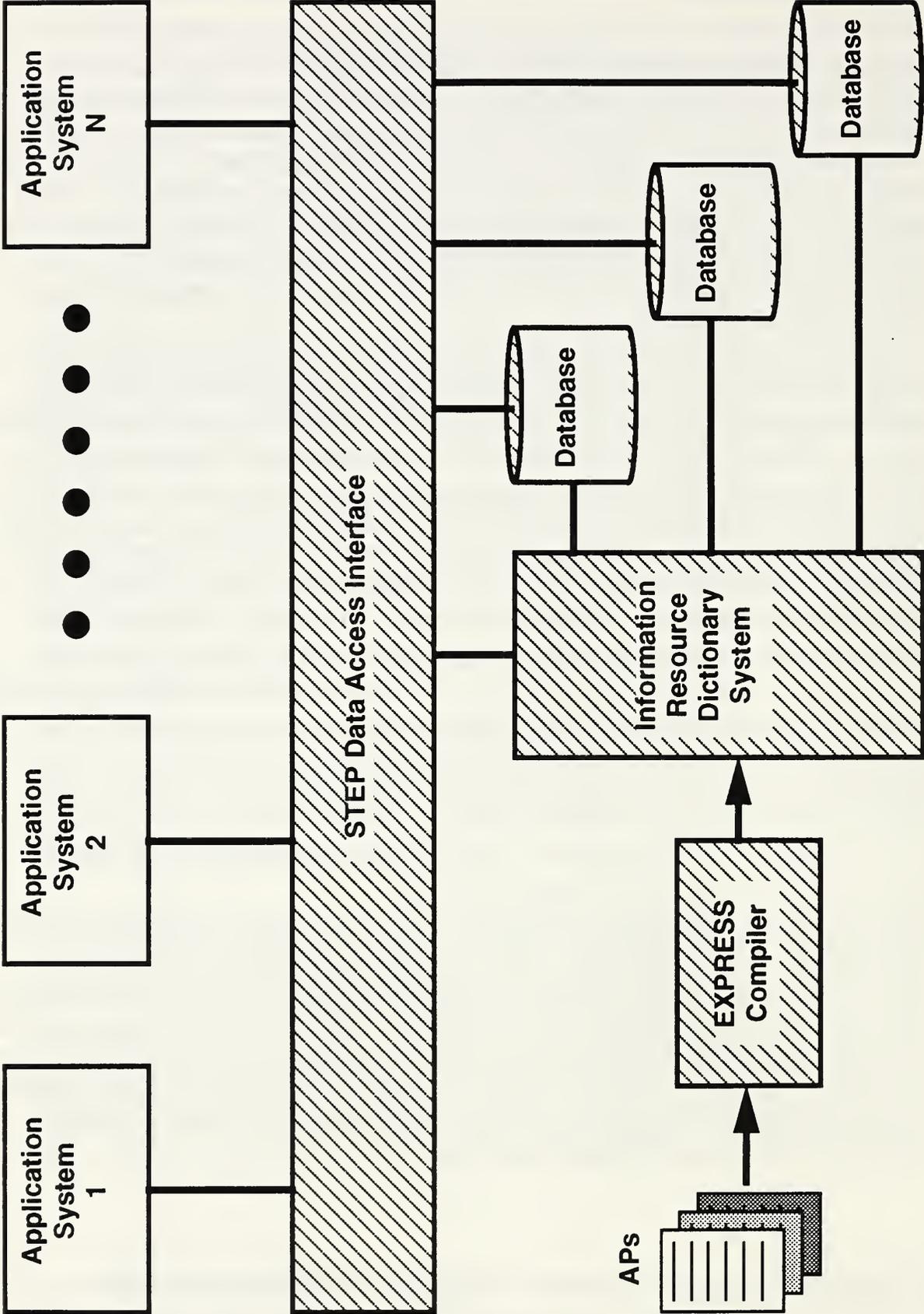


Figure 8: Shared Database Components

Non-STEP data is also stored in the Data Repository Subsystem. Storage of and access to such data may be different than for STEP data. We could make the simplifying assumption that the data in question is (or can be) modeled in EXPRESS. Using this assumption everything described above for STEP data would hold true for non-STEP data as well. Describing it in EXPRESS would permit it to be stored in the repository through the EXPRESS compiler/IRDS combination and accessed using the SDAI. The alternative is somewhat more complicated.

Storage of non-STEP data in the repository could be accomplished in several ways. A particular database within the repository could be set aside particularly for this purpose and access to the data could be accomplished through whatever interface was supplied for the selected database. Another method, employing the IRDS, would require direct access to the IRDS service functions to manage and access the data within the repository's databases. Obviously the method chosen for managing and accessing non-STEP data directly impacts the implementation of the shop floor translation software described for the Manufacturing and Inspection Engineering Subsystems and the production interface translator in the Shop Floor Subsystem.

3 STEP Implementation Considerations

This section provides some insight regarding the implementation prospects resulting from the use of STEP in the architecture described in the previous section. Bear in mind that the aspects of STEP which will primarily impact implementation of the architecture's logical systems are the APs defining the information exchange between systems, the EXPRESS information modeling language, the STEP implementation specifications supported (e.g., exchange file specification, SDAI), and the STEP conformance criteria[†] ultimately defined. We now look at three implementation scenarios; one which could use existing systems, one which could use near-term systems, and one which employs what we speculatively refer to as next generation systems.

3.1 Existing Systems

Current commercial CAD/CAM systems typically provide facilities for data exchange using IGES. As described in the introduction, such data interchange takes place using file exchange. While STEP is certainly more complex than IGES, it should not be long before commercial CAD/CAM vendors begin to support STEP data exchange via files. Thus we can envision how the applications in this architecture - the Design, Manufacturing, and Inspection Engineering Subsystems - would be implemented in a STEP environment.

Whatever the content of the APs any of the application systems implement, it is unlikely that there will be a one-to-one relationship between the data representations internal to an application and the data representations specified in an AP. STEP translation software, which transforms the application's internal data representations into those required by the APs (and vice versa), will be needed for each application. Such translation software, referred to as pre- and post-processing software in section 1, was identified as the STEP interface subsystem for the discussions in section 2.

In order to develop such a translator for STEP, a software developer must understand and have software access to the internal data representations the application already uses. The software developer must also understand the contents of the AP to be supported and the STEP exchange file specification. Understanding an AP requires knowledge of the information domain the AP covers and of the information modeling language EXPRESS. Understanding the exchange file specification also requires knowledge of EXPRESS.

Implementing the transformation between the application's internal data representations and the AP's data representations can be accomplished by encoding the transformation directly, or indirectly using supplementary mapping software. Conformance testing will determine whether the translator can produce STEP exchange files containing data according to the AP's intent and whether the translator interprets such files correctly. If the translator for the application successfully completes these conformance tests, the application supports the AP.

[†] STEP conformance criteria, tests, and procedures for testing are all still evolving.

We made the point in section 2.4 that when exchanging STEP data via files, the STEP repository can be made arbitrarily simple. Essentially, the collection of STEP data files representing product descriptions is the repository. With Design, Manufacturing, and Inspection Engineering Subsystems providing the functionality described in section 2 and implemented as described here, the architecture is realized.

3.2 Near-Term Systems

In this implementation scenario, the previous discussion of existing systems is adapted to work in the data sharing environment described in section 2.5. From the application system's perspective, the only change is how the translation software is implemented. Instead of producing and consuming STEP data in the exchange file format, the translator now produces and consumes STEP data according to the SDAI specification. Modifying a STEP file translator, or developing a new translator, to work with SDAI requires knowing how to use the SDAI but still requires interpretation of APs described in EXPRESS. Prototypes of both software which implements SDAI-like functionality and systems which use those interfaces have been developed [Cla91]. On the basis of such prototypes and the evolution of the SDAI specification, we could expect to see commercial realization and usage within three years.

The conformance tests for an application using SDAI are somewhat more complex than for a file exchange environment. The tests for conformance will first have to ascertain that the application's translator uses the SDAI correctly and then determine whether the application produces and interprets STEP data using SDAI according to the intent of the APs.

From the Data Repository perspective, the situation is considerably different than that described for a file exchange environment. Product descriptions are not stored in STEP exchange files - they are now stored in the Data Repository software. Access to the Data Repository, and therefore to product descriptions, is provided by the SDAI. Database vendors may take it upon themselves to implement the SDAI over their database product, as is the case with SQL [FIP90] implementations. On the other hand, third party developers may perform system integration of software supporting IRDS, multiple database products, and implement the SDAI over these different components. In either case, the implementer of an SDAI must certainly understand the specification itself and understand EXPRESS. While an implementer may not need to understand the domain of information covered by APs the Data Repository supports, the implementer may be able to fine-tune the performance of the Data Repository if given a reasonable understanding of the AP information and its usage characteristics.

At the least, Data Repository Subsystems will have to be tested to ensure that they conform to the SDAI specification. It is also possible that they will need to be tested for AP support as well, in a fashion similar to that for applications. Whether or not this will be necessary will become clearer as the details of SDAI are resolved in the STEP community.

The architecture is realized in this scenario through establishment of the SDAI specification, modification of the applications to work with SDAI, and by implementation of the underlying Data Repository Subsystem.

3.3 Next Generation Systems

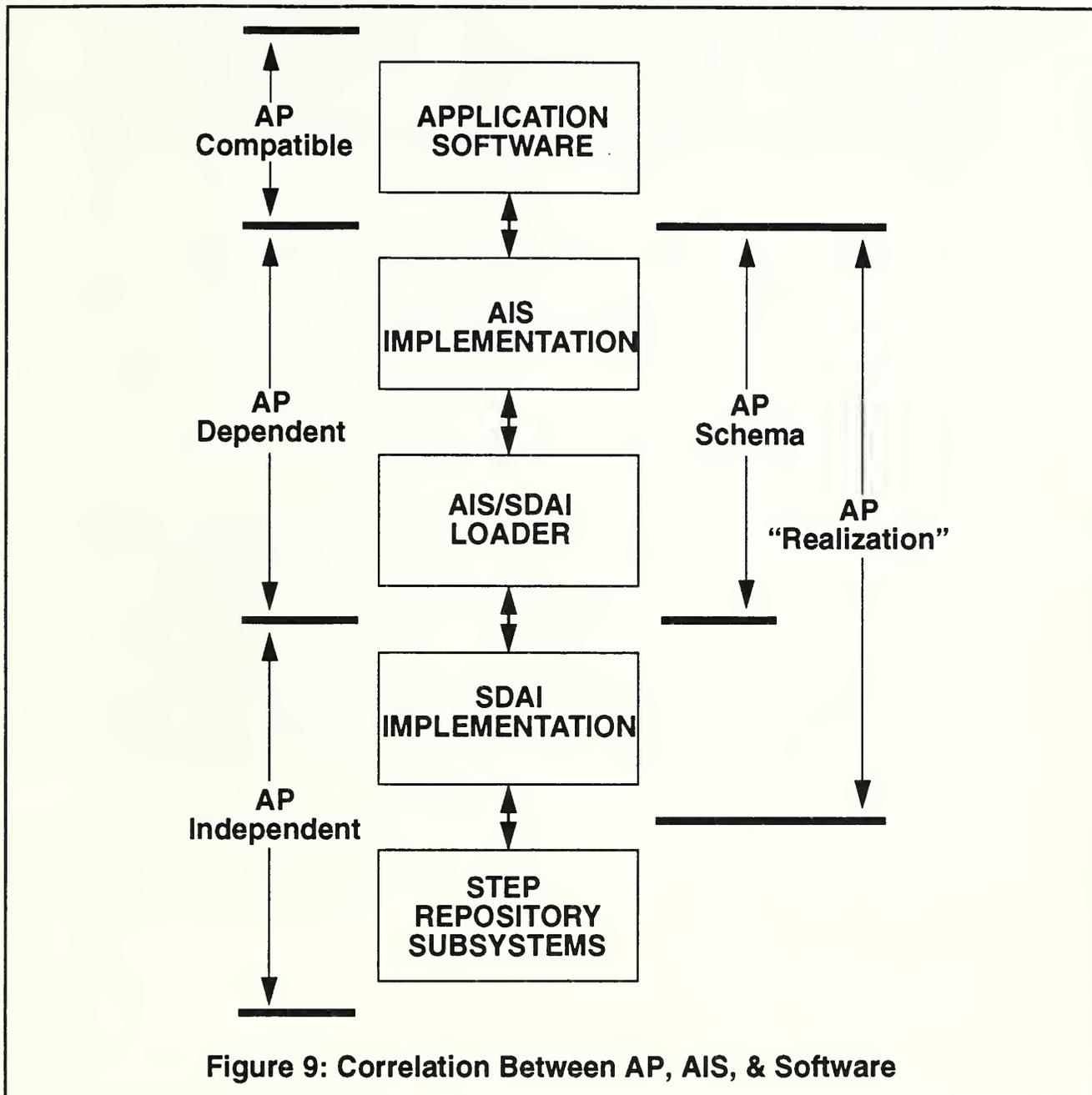
In this section we consider a realization of the architecture which builds on the implementation described in section 3.2. This implementation includes another interface layer, a layer which we believe directly corresponds to the intended use of APs. First, some background information on the CAM-I Application Interface Specification (AIS) [Mag91] is necessary.

The AIS is an emerging standard that is intended to complement the STEP effort. The AIS addresses standardization of the programming interfaces to product modeling systems. The AIS effectively surrounds a modeler providing a standardized virtual modeler to application programs. This standardized virtual modeler is based on a STEP data model: the current scope of the AIS is solid geometric modeling including both boundary representation and constructive solid geometry from Part 42[ISO42]. The AIS concept exceeds the current scope of STEP by normalizing functionality associated with the data, i.e., it specifies the manipulation of STEP data entities in the context of a modeling system for those entities. For clarity, the currently defined AIS shall be referred to as the SM-AIS (Solid Modeling - Application Interface Specification).

Given that the SM-AIS addresses a specific domain in a particular context, it is natural to infer a correlation between the SM-AIS and an AP which applies solid geometric modeling representations to product description[†]. The general concept of an AIS could be thought of as implicitly defining a portion of an AP, e.g., how the data associated with an AP is to be used and manipulated. Tying the general AIS concept to that of APs would require extensions to the information modeling capabilities of EXPRESS - in fact the ISO committee responsible for EXPRESS development is already considering such extensions. The idea of an AIS reflecting a particular AP could be used as a paradigm for development of AIS's in addition to the current SM-AIS.

[†] [ISO204] currently proposes the use of boundary representation for mechanical product definition. This AP excludes constructive solid geometry.

Consider how supplementing APs with the AIS concept could influence the implementation of application systems. Figure 9 shows the relationship between



software, an AP, and an AIS. Previously we described how the application software interacted with the Data Repository Subsystem through the SDAI. Now two modules provide intermediate functionality between the application and SDAI software. The AIS implementation embodies the manipulations provided against the data according to a particular AP. These manipulations are referred to as "AP-Dependent" (entity-specific) operations since they are tailored to the data and its context as specified by an AP. The AIS implementation may be a very complex piece of software; for the SM-AIS this module essentially provides the capabilities of a solid geometric modeler. The AIS/SDAI loader performs the

function of moving data according to the requirements of the AIS implementation across the SDAI implementation. Since the loader is tailored for a particular AIS it too is "AP-Dependent." The functionality of the SDAI is not tailored to a specific AP - therefore we refer to it as "AP-Independent" (or entity-independent). One can think of the AIS/SDAI loader as playing a role similar to that of the translators described in section 3.2. The collection of the AIS implementation, the AIS/SDAI loader, and the SDAI implementation can be thought of as together realizing an AP and presenting AP-compatible capabilities to application software.

We can also see how application software incorporating AIS functionality for an AP could be more thoroughly tested for conformance. Semi-automated conformance testing software would have two "test points" in the application. One, where the application makes use of the SDAI, would permit tests of the application's ability to produce and consume data in accordance with the constraints described by the AP. The second, where the application makes use of the AIS, would permit tests of the application's ability to manipulate data in accordance with the AP's intended interpretation.

Let's return to the specific case of the applications described in this architecture. It should be clear that the STEP interface subsystem described for each of the three applications would be realized by the combination of AIS/SDAI loader and AIS implementation shown above. Although prototypes of the SM-AIS have been developed [Gun91] they have not made use of an underlying STEP Data Repository Subsystem since the SDAI is still evolving. The timeframe to realize this type of implementation, requiring establishment of SDAI, the underlying repository subsystems, and AIS's for the APs of interest, is definitely greater than three years.

4 Summary

This document has presented a conceptual architecture for a mechanical parts production system using STEP as the primary means for product information exchange between software applications. We have discussed the functionality of the major systems and discussed the internals of those systems as they relate to the use of STEP. The major focus has been on the software components required to make use of STEP in the context of mechanical parts production. Finally, we have offered insight as to how the software components could be implemented both now and in the future.

We have identified that STEP Application Protocols - APs - will define what information is interchanged between the engineering applications of the system. These APs have been designated in an abstract way as the Manufacturing and Inspection Interface APs. We use these designations as place-holders: the information described by these two APs may be realized in the standards community as several APs[†]. Nevertheless the concepts described for the software components and their implementation should remain applicable.

This architecture can serve as a guidebook for those intending to implement an MPPS or related system. With more APs becoming available, implementation of such systems can be considered. A real implementation of a system would require selection of the APs appropriate to the product domain and engineering and fabrication capabilities. Choices would have to be made when designing the implementation: whether to use STEP data internally within application subsystems, whether to employ file exchange or shared database, how sophisticated a data repository to employ, and how to handle non-STEP data. These issues can, and should, be addressed incrementally in implementations. The benefit of such implementations would not be diminished. Starting work now on STEP implementations will provide valuable experience for software developers while the software users will see real progress towards removing barriers to productivity.

[†] In fact, as this document goes to print, there are proposals to the STEP standards bodies for work on two new APs: one for Numerical Controlled Processes for Machined Parts, and the other for Process Plans for Machined Parts. These APs would be completely applicable to the information domain described in this document as the Manufacturing Interface AP.

A Glossary

AIS

Application Interface Specification; a proposed US standard specifying a software interface to product modelers.

Application Protocol(s)/AP(s)

A specification of a subset of STEP data, the context of this data, and the usage of this data for the purposes of meaningful exchange between particular applications.

APT

Automatically Programmed Tools; a task oriented language used for directing numerically controlled machines tools.

BCL

Binary Cutter Location; an exchange format for conveying instructions to NC machine tools.

CAD

Computer-Aided Design; software used by designers and engineers to produce a computer representation of a product, part, assembly, structure, etc.

CMM

Coordinate Measuring Machine; a computer-driven machine which can be directed to take measurements on a part.

IGES

Initial Graphics Exchange Specification; an existing standard used largely for exchanging the computer representations of engineering drawings between Computer-Aided Design systems.

IPO

IGES/PDES Organization; the voluntary organization in the US devoted to development of IGES and STEP.

IRDS

Information Resource Dictionary System; a specification for software to be used for management of complex data systems.

ISO

International Organization for Standardization; the international voluntary organization devoted to developing and setting standards - STEP is just one of the many standards this organization is responsible for.

MPPS

Mechanical Parts Production System; the manufacturing system this architecture addresses.

NC

Numerical Control; a historical term now generally used to mean the programs or means of controlling manufacturing equipment via computer.

NPT

National PDES Testbed; the NIST facility devoted to development, testing, and dissemination of STEP.

PDES

Product Data Exchange using STEP; the US efforts toward the development of STEP.

PDES, Inc.

An expanding consortium of companies formed in 1988 for the purpose of accelerating the development and use of STEP.

SQL

Structured Query Language; a software language designed for the specification and manipulation of information in a relational database.

SDAI

STEP Data Access Interface; an evolving specification describing a STEP implementation mechanism.

STEP

Standard for the Exchange of Product Model Data; it is the proposed international standard for product representation and exchange.

B References

- [ANS87] "American National Standards - for information systems - programming language - APT", American National Standards Institute, New York, NY, 1987.
- [ANS90] ATIS, Information Resource Dictionary System, ANSI X3H4 Working Draft, February 1990.
- [Cla91] Clark, S., et. al., "Prototype Evaluation for STEP Data Access Interface Prototype Implementation", National Institute of Standards and Technology, Document in Preparation.
- [EIA83] "RS-494 32 Bit Binary CL (BCL) Input Format for Numerically Controlled Machines", Electronic Industries Association, Washington, DC, 1983.
- [FIP90] "Database Language SQL", FIPS Publication 127-1, February 1990. Available from National Technical Information Service.
- [Fow90a] Fowler, J.E., "STEP Production Cell Technical Development Plan", NISTIR 4421, National Institute of Standards and Technology, Gaithersburg, MD, September 1990.
- [Fow90b] Fowler, J.E. ed., "STEP Data Access Interface Specification", ISO TC184/SC4/WG7 Document N499, August 1990.
- [Fre90] Frechette, S., Jurrens, K., "Development Plan: Product Data Exchange Network", NISTIR 4431, National Institute of Standards and Technology, Gaithersburg, MD, September 1990.
- [Gun91] Gunn, K., et. al., "Implementing the AIS Version 2.0 with Parasolid", Research Report to the Product Modeling Program of CAM-I, Arlington, TX, Report Number to be determined, September 1991.
- [ISO1] ISO 10303-1 Overview and Fundamental Principles.[†]
- [ISO11] ISO 10303-11 Description Methods: The EXPRESS Language Reference Manual.[†]
- [ISO21] ISO 10303-21 Clear Text Encoding of the Exchange Structure.[†]

[†] This Part is one in a series of Parts which together comprise the International Standard ISO 10303 Industrial Automation Systems - Product Data Representation and Exchange.

- [ISO31] ISO 10303-31 Conformance Testing Methodology & Framework: General Concepts.[†]
- [ISO32] ISO 10303-32 Conformance Testing Methodology & Framework: Requirements on the Testing Laboratory and the Client for the Conformance Assessment Process.[†]
- [ISO42] ISO 10303-42 Integrated Generic Resources: Geometric and Topological Representation.[†]
- [ISO45] ISO 10303-45 Integrated Generic Resources: Materials.[†]
- [ISO47] ISO 10303-47 Integrated Generic Resources: Shape Tolerances.[†]
- [ISO48] ISO 10303-42 Integrated Generic Resources: Form Features.[†]
- [ISO203] ISO 10303-203 Application Protocol: Configuration Controlled Design.[†]
- [ISO204] ISO 10303-204 Application Protocol: Mechanical Design Using Boundary Representation.[†]
- [ISO205] ISO 10303-205 Application Protocol: Mechanical Design Using Surface Representation.[†]
- [Kem91] Kemmerer, S.J., "Development Plan: STEP Conformance Testing Service", NISTIR 4641, National Institute of Standards and Technology, Gaithersburg, MD, August 1991.
- [Kra91] Kramer, T., Palmer, M., Feeney, A.B., "Issues and Recommendations for a STEP Application Protocol Framework", National Institute of Standards and Technology, Draft, September, 1991.
- [Mag91] Magleby, S., Jackson, D., "A Standardized Application Interface for Geometric Modelers", Product Modeling for Computer-Aided Design, selected papers from the 3rd IFIP Workshop on Geometric Modeling, Rensselaerville, NY, June 1990, Elsevier Science Publishers, The Netherlands, 1991.
- [Pal91] Palmer, M., Gilbert, M., Anderson, J., "Guidelines for the Development and Approval of STEP Application Protocols", National Institute of Standards and Technology, Gaithersburg, MD, Draft, July 1991.

[†] This Part is one in a series of Parts which together comprise the International Standard ISO 10303 Industrial Automation Systems - Product Data Representation and Exchange.

References

- [PDE91] PDES, Inc., "A High-level Architecture for Implementing a PDES/STEP Data Sharing Environment", PTI017.03.00, South Carolina Research Authority, Charleston, SC, May 1991.
- [Ree90] Reed, K., ed., "The Initial Graphics Exchange Specification (IGES) Version 5.0", NISTIR 4412, National Institute of Standards and Technology, Gaithersburg, MD, September 1990.
- [Sen91] Senehi, K., et. al., "Manufacturing Systems Integration Initial Architecture Document", National Institute of Standards and Technology, Draft, September 1991.

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER	NISTIR 4685
2. PERFORMING ORGANIZATION REPORT NUMBER	
3. PUBLICATION DATE	SEPTEMBER 1991

4. TITLE AND SUBTITLE
A Conceptual Architecture for a Mechanical Parts Production System Based on STEP.

5. AUTHOR(S)

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS) U.S. DEPARTMENT OF COMMERCE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY GAITHERSBURG, MD 20899	7. CONTRACT/GRANT NUMBER
	8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Office of the Secretary of Defense
CALs Program Office
Pentagon
Washington, DC 20301-8000

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Standard for the Exchange of Product Model Data (STEP) is an emerging standard addressing the problems of data exchange and representation of produced goods in a variety of manufacturing enterprises. Given that the initial STEP specifications largely pertain to mechanical parts production, this domain is an appropriate context for initial STEP implementations. This document describes an architecture for systems realizing a mechanical parts production capability using STEP data exchange. The functions of the major systems and relationships between systems is discussed. Software components which could be used to implement major systems are identified. Major emphasis is given as to how STEP is implemented and used in the context of the architecture.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
PDES; STEP; CAD/CAM; STEP Application Protocols; STEP Implementation; AIS; SDAI

13. AVAILABILITY	14. NUMBER OF PRINTED PAGES
<input checked="" type="checkbox"/> UNLIMITED FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).	38
<input type="checkbox"/> ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.	15. PRICE
<input checked="" type="checkbox"/> ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.	A03

